APPLICATION FOR UNITED STATES PATENT

INVENTORS: Daniel P. Starkovich
Scott L. Rappa
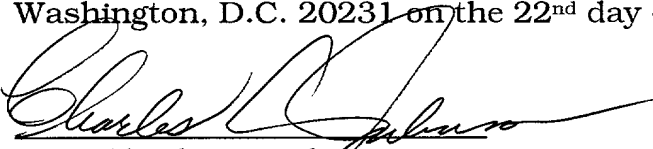Joey L. Erickson

INVENTION: PASS THROUGH - XOCTET

**DOCKET
NUMBER:**   RA 5342 (33012/303/101))

Unisys Corporation
Charles A. Johnson
P O Box 64942 - MS 4773
St. Paul, MN 55164
Attorney for Applicant
Reg. No.: 20,852

SPECIFICATION

CERTIFICATE UNDER 37 CFR 1.10: The undersigned hereby certifies that this transmittal letter and the paper of papers, as described hereinabove, are being deposited in the United States Postal Service, "Express Mail Post Office to Addressee" having an Express Mail mailing label number of EJ359881317US, in an envelope addressed to: ASSISTANT COMMISSIONER OF PATENTS, Washington, D.C. 20231 on the 22nd day of December, 2000.

_____

Charles A. Johnson

PASS THROUGH - XOCTET

CROSS REFERENCE TO CO-PENDING APPLICATIONS

5   The present application is related to U.S. Patent Application Serial No. XX/XXX,XXX, filed _____, entitled "A GENERIC DCOM SERVER; U.S. Patent Application Serial No. XX/XXX,XXX, filed _____, entitled "A MULTI-USER CUSTOMIZED DCOM GATEWAY FOR AN OLTP ENTERPRISE SERVER APPLICATION"; U.S. Patent Application Serial No. XX/XXX,XXX,

10   filed _____, entitled "WEBTX MESSAGE QUEUE SYSTEM"; U.S. Patent Application Serial No. XX/XXX,XXX, filed _____, entitled "WEBTX TRANSACTION QUEUE SYSTEM"; U.S. Patent Application Serial No. XX/XXX,XXX, filed _____, entitled "Adapter Architecture; U.S. Patent Application Serial No. XX/XXX,XXX, filed _____, entitled "XGate"; U.S.

15   Patent Application Serial No. XX/XXX,XXX, filed _____, entitled "Java RMI Adapter"; and Application Serial No. XX/XXX,XXX, filed _____, entitled "A COMMON GATEWAY WHICH ALLOWS APPLETS TO MAKE PROGRAM CALLS TO OLTP APPLICATIONS EXECUTING ON AN ENTERPRISE SERVER"; which are assigned to the assignee of the present

20   invention and incorporated herein by reference.

BACKGROUND OF THE INVENTION

25   **1. Field of the Invention** - The present invention relates to methods and apparatus for facilitating on-line processing requests, and more specifically, to adapting client formats to the use of a gateway in communicating with an On-Line Transaction Processing (OLTP) Enterprise Server from user work stations.

2. <u>Description of the Prior Art</u> - The methods by which companies conduct business with their customers are undergoing fundamental changes, due in large part to World Wide Web technology. In addition, the same technology that makes a company accessible to the world, may be

5   used on internal company networks for conducting operational and administrative tasks.

One of the technologies underlying the World Wide Web is the prospect of using component software technology -- the idea of breaking large, complex software applications into a series of pre-built and

10   easily developed, understood, and changed software modules called components -- as a means to deliver software solutions much more quickly and at a lower cost (source: DCOM: A Business Overview, online at http://www.microsoft.com/ntserver/guide/dcom.asp). The goal is to achieve economies of scale for software deployment across the industry.

15   A component architecture for building software applications will enable this by: 1) speeding development - enabling programmers to build solutions faster by assembling software from pre-built parts; 2) lowering integration costs - providing a common set of interfaces for software programs from different vendors means less custom work is required to

20   integrate components into complete solutions; 3) improving deployment flexibility - making it easier to customize a software solution for different areas of a company by simply changing some of the components in the overall application; and 4) lowering maintenance costs - isolating software function into discreet components provides a low-cost, efficient mechanism

25   to upgrade a component without having to retrofit the entire application.

A distributed component architecture applies these benefits across a broader scale of multiuser applications. The Distributed Component Object Model (DCOM), developed by Microsoft Corporation, has several

strengths that make it a key technology for achieving this. Because it is an ActiveX technology, DCOM works natively with Internet technologies like TCP/IP, the Java language, and the HTTP network protocol, providing "object glue" that will enable business applications to work across the Web.

5   DCOM is also an open technology that runs on multiple platforms.

DCOM has its roots in Microsoft's object technology, which has evolved over the last decade from DDE (Dynamic Data Exchange, a form of messaging between Windows programs), OLE (Object Linking and Embedding, embedding visual links between programs within an

10   application), COM (the Component Object Model, used as the basis for all object binding), and ActiveX (COM enabled for the Internet).

As stated earlier, applications built from components are simply easier to debug and evolve than large, monolithic applications.

The logical boundary for component applications is no longer on a

15   single machine. Businesses want to leverage the benefits of component development across a broader set of shared applications that operate on multiple machines. These types of applications are referred to as "three-tier" or "n-tier" applications, where "tiers" of application logic, presentation services, business services, and information retrieval and management

20   services, are broken into different components that can communicate directly with each other across a network. To the end user, these applications appear as a seamless extension of their existing desktop environment.

The simplicity, ubiquity, and industry momentum of standard

25   Internet protocols like HTTP make it an ideal technology for linking components together for applications that span machine boundaries. HTTP is easy to program, is inherently cross-platform, and supports an accessible, universal naming service. Much of the excitement around the

Java language derives from its potential as a mechanism to build distributed component applications on the Internet. In addition to Java support, DCOM enables components written in other languages, including C, COBOL, Basic, and Pascal, to communicate over the Internet, providing

5   a growth path for existing applications to support Web technology.

As distributed component architectures, such as DCOM, are making their mark as a technology that enables software components to communicate directly with each other across networks, many businesses have a wealth of information that is managed by prior art data base

10   management systems such as DMS, RDMS, DB2, Oracle, Ingres, Sybase, Informix, and many others. In addition, many of the database management systems are available as resources in a larger transaction processing system.

One key to the future success of a business may lie in its ability to

15   capitalize on the ability to interconnect a distributed component architecture, such as DCOM, with existing enterprise On-line Transaction Processing (OLTP) systems. It defeats the two main goals of component-based development, fast time-to-market and lower development costs, if companies are forced to "hand code" into their component applications the

20   mission critical services that are required for online production systems. Therefore, the leading system suppliers have developed commercially available "middleware" to link web based work stations with existing OLTP systems.

However, most existing OLTP systems have been developed under the

25   assumption that user work stations are physically, electrically, and functionally dedicated exclusively to providing communication between the OLTP and a single user during an entire user session period. This assumption arose at a time in which user work stations were simply dumb

4

video display/keyboard devices connected directly to the OLTP mainframe via a dedicated electrical line.

Modern work stations, however, are extremely complex and capable of substantial unassisted data processing. Furthermore, the internet connection between a modern work station and the OLTP enterprise system is anything but physically, electrically, and functionally dedicated exclusively to a single user session.

A further problem arises with the current formats of the potential work stations requesting service. The common solution is to provide a separate gateway for each of the disparate formats encountered. However, the development of a large number of gateways is costly and time consuming. Furthermore, there is not any provision for employing the XML message format with legacy data base management systems.

A current format not accommodated by any legacy mainframe data base management systems is Java Remote Method Invocation (RMI). This is a very popular format not supported for current mainframe operations.

An approach to the creation of an interface from an Internet work station to a mainframe legacy data base management system is through the preparation of predefined view buffers. Through this technique, one or more data formats are established within the middleware necessitating two conversion stages (i.e., work station to view buffer and view buffer to legacy data base management system).

SUMMARY OF THE INVENTION

The present invention overcomes many of the disadvantages associated with the prior art by providing a method and apparatus for

5   passing service requests and data from web based work stations directly to OLTP enterprise server systems with or without the use of view buffers. In the preferred mode, the work station is an industry compatible personal computer running a commercially available browser operating under a common operating system such as Windows. The client work station is

10   coupled, via the internet, to an NT server through the middleware product of the present invention which is a generic WebTx gateway. This gateway permits the user work station to communicate as with other type internet applications.

A plurality of adapters convert each of the differing request formats

15   into a single input format for the single generic gateway. In this manner, a client may make a service request via C++, ASP (i.e., an active server page from a web browser), or other format. The appropriate corresponding adapter converts the service request to a standardized format and presents the information to the generic gateway. After processing, the generic

20   gateway couples service request information to one of a plurality of corresponding connectors for transfer to the end service provider.

In accordance with one preferred mode of the present invention, a service request is made from an industry compatible personal computer in the form of an ASP created by a commercially available web browser. The

25   web page is transferred to the ASP adapter and converted into the standardized form. The converted service request is transferred to the generic gate, which sends the information to the appropriate one of the

plurality of connectors. The selected connector transfers the service request to an enterprise server which honors the service request.

The use of a plurality of adapters and a plurality of connectors, in combination with a single generic gateway, greatly simplifies the development process in that new gateways need not be generated each time a new service request format is encountered. The adapters may be created and added to the middleware system in a "plug and play" manner. Each new adapter only needs to communicate with the standardized generic gateway input. Because the adapters are much more easily generated than completely new gates, significant development savings are realized.

In accordance with one embodiment of the present invention, access to mainframe services is available using XML (Extensible Markup Language) formatted data. XML is quickly becoming the middleware-independent message interchange format of choice for business-to-business and business-to-customer Internet commerce. A client is now able to use this standards based, self-describing message format to access mainframe transactions.

This embodiment permits any of the adapter client types to access mainframe transactions with the XML message format. The XML message does not need to be described by a view. This buffer is then passed to the mainframe transaction through any of the available connectors. The transaction that is executed on the mainframe returns a data buffer that is also not described by a view. XGate takes the buffer and passes it to the client application.

In an alternative embodiment, the present invention provides a technique for gaining access to mainframe services using Java Remote Method Invocation (RMI) as the underlying object architecture. A Java RMI client is now able to communicate and access mainframe transactions from

a remote site. For example, a person at a Java based end user workstation can perform transactions on an existing mainframe database. Accessing the transactions is established through a Java RMI adapter. This adapter can be used to communicate with GGate and XGate using view based or

5   non-view based data in GGate or XML formats. A client, using RMI protocol, sends the adapter information regarding the transaction or service the client wishes to perform. The adapter passes information to the gate on a Java server. The gate then interfaces with the connector and finally the service is performed on the mainframe.

10   Previously, GGate (and all of Transaction Integrator (TxI))dealt with view-based data for every request. TxI performed appropriate data parsing on input and packaged the data into buffers based on a view's format. On the response side, TxI got a buffer back and populate the results based on the output view's format. This, in some cases, put a requirement on users

15   to fit their data into defined views. Therefore, in accordance with the present invention, Ggate, Xgate, and their adapters (out-of-process DCOM, in-process DCOM, CORBA, and Java RMI) support pass through requests. Pass through requests use non-view based data. Thus, users may pass data via TxI without creating the views first.

BRIEF DESCRIPTION OF THE DRAWINGS

Other objects of the present invention and many of the attendant advantages of the present invention will be readily appreciated as the same

5   becomes better understood by reference to the following detailed description when considered in connection with the accompanying drawings, in which like reference numerals designate like parts throughout the figures thereof and wherein:

FIG. 1 is a functional block diagram of an exemplary computing

10   environment;

FIG. 2 is a functional block diagram of a generalized embodiment;

FIG. 3 is a more detailed functional block diagram of the generalized environment;

FIG. 4 is a block diagram of a typical environment employing the

15   present invention;

FIG. 5 is a functional block diagram of components utilized within a Microsoft NT environment;

FIG. 6 is a diagram showing the relationship of the key run time software components of the present invention which allow clients to access

20   enterprise applications;

FIG. 7 is a flow diagram illustrating what happens when a request is issued by a client to an enterprise OLTP enterprise application;

FIG. 8 is a block diagram showing operation of Xgate; and

FIG. 9 is a block diagram showing of the present invention with both

25   Ggate and Xgate environments.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The detailed descriptions which follow are presented largely in terms of algorithms and symbolic representations of operations on data bits within

5   a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art.

An algorithm is here, generally, conceived to be a self-consistent sequence of steps leading to a desired result. These steps are those

10   requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements,

15   symbols, characters, terms, numbers or the like. It should be kept in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

Furthermore, the manipulations performed are often referred to in

20   terms, such as adding or comparing, which are commonly associated with mental operations performed by a human operator. No such capability of a human operator is necessary, or desirable in most cases, in any of the operations described herein which form part of the present invention; the operations are machine operations. Useful machines for performing the

25   operations of the present invention include general purpose digital computers or other similar devices. In all cases, it should be kept in mind the distinction between the method operations in operating a computer and the method of computation itself. The present invention related to method

10

steps for operating a computer in processing electrical or other (e.g., mechanical, chemical) physical signals to generate other desired physical signals.

The present invention also relates to apparatus for performing these

5　operations. This apparatus may be specially constructed for the required purposes or it may comprise a general purpose computer as selectively activated or reconfigured by a computer program stored in the computer. The algorithms present herein are not inherently related to a particular computer system or other apparatus. In particular, various general

10　purpose computer systems may be used with computer programs written in accordance with the teachings of the present invention, or it may prove more convenient to construct more specialized apparatus, to perform the required method steps. The required structure for such machines will be apparent from the description given below.

FIG. 1 is a functional block diagram of an exemplary computing environment in which the present invention could be used to make an enterprise based transaction processing system interoperable with a PC/Workstation based requestor employing the Java RMI architecture. A

5 plurality of PC/Workstations, designated as Clients 10, 12, 14 and 16 are coupled to a Server 18 via Network 20. The Network 20 may be an internal local area network or the Internet.

Each of the Clients 10, 12, 14 and 16, is a Personal Computer/Workstation having operating system software and application

10 software designed to provide Graphical User Interface (GUI) and communications capabilities which enable the Client to communicate with an associated Server application 18 via a Network 20.

The Workstation Server System 50 may be any class of machine(s) which are capable of running a Server application 18 along with a

15 Distributed Transaction Processor 54. The Transaction Processing system 54 is designated as *Distributed* to make clear that a transaction is formatted on the Workstation Server System 50 and forwarded to the Enterprise Server system 52 for processing. As is explained in further detail below, Distributed Transaction Processing System 54 contains the Queued

20 Transaction connector and the input and output queues required to practice the preferred mode of the present invention.

The exemplary Enterprise Server System 52 is a 2200 Series data processing system from Unisys and also includes a Distributed Transaction Processing System 56. The Distributed Transaction Processing System 56

25 is intended to encompass the same functionality as a monolithic transaction processing system, however, it is designated as *Distributed* to be compatible with the Distributed Transaction Processing System 54. The exemplary Distributed Transaction Processing Systems 54 and 56 are

intended to encompass transaction manager software, such as Open/OLTP Transaction Manager software from Unisys, and user implemented Open/OLTP services. The Distributed Transaction Processing System 54 and the Distributed Transaction Processing System 56 are coupled via

5   Network 58. Preferably, the network interface for Network 58 is separate from the network interface for Network 20.

The Distributed Transaction Processing System 56 serves data from the Database 28 to the Transaction Clients 30, 32, 34 and 36. The Transaction Clients 30, 32, 34 and 36 are coupled to the Distributed

10   Transaction Processing System 56 via line 38, of which the underlying technology is driven by the application of the Distributed Transaction Processing System 56.

The Transaction Gateway Client 40 allows the Server 18 to interoperate with the Transaction Processing System. When a Client 10, 12,

15   14 or 16 selects an enterprise based service, the request is routed to the Server 18, which in turn routes the request to the Transaction Gateway Client 40. The Transaction Gateway Client 40 determines the requested service and forwards the necessary information to the Distributed Transaction Processing System 54 and 56. The Distributed Transaction

20   Processing System 54 and 56 processes the request against the Database 28 according to the specified request (e.g., select, update, delete). The Distributed Transaction Processing System 54 and 56 returns data and/or status information to the Transaction Gateway Client 40, which in turn formats the data in an appropriate manner for the Server 18. The Server 18

25   then returns the information to the requesting Client 10, 12, 14 and 16.

FIG. 2 is a functional diagram of the major components of the generalized environment. In this example, DCOM client 68 generates a service request using visual basic (i.e., VB), C++, active server page (ASP),

13

or other suitable tools. The service request is transferred to DCOM Server (adapter) 70. As explained in detail below, there is a different adapter for each input format employed.

The adapter of DCOM Server 70 converts the service request input to the standardized input utilized by GGATE 72. This generic gateway is described in greater detail in the above incorporated, co-pending, and commonly assigned patent applications. After processing, GGATE 72 routes the service request via the appropriate of the plurality of Connector 0 73, Connector 1 74, or Connector 2 75. The selected connector transfers the service request to End Service Provider 78, which honors the service request.

Fig 3 is a functional block diagram of a populated generalized embodiment showing a plurality of clients, adapters, and connectors. Client 0 92 prepares service requests in a first format, such as ASP. Client 1 90 prepares service requests in a second format, such as visual basic. Client m 94 prepares service requests in a third format.

Each of the respective clients sends its service requests to the respective associated adapter as shown. Client 0 92 service requests are sent to Adapter 0 84. Client 1 90 service requests are sent to Adapter 1 86, and Client m 94 service requests are sent to Adapter m 88. As explained above, each of these adapters (i.e., 84, 86, and 88) converts the inputs to a single format for input to GGATE 72.

The operation of GGATE 72 is explained in detail in the above cross-referenced U.S. Patent Applications. The converted and process service requests are transferred to the appropriate one of Connector 0 73, Connector 1 74, or Connector m 75. These connectors are arranged to handle the different formats as with the plurality of adapters. The

converted, processed, and connected service requests are transferred to End Service Provider 78 for honoring.

FIG. 4 is a functional block diagram of a generalized environment suitable for incorporating the present invention. In general, this is

5   middleware in a client/server computing environment which accepts requests from the client side and routes the requests to the correct place on the server side, then passes a response from the server side back to the client side. In the context of the present invention, a generic gate "marries" a Microsoft DCOM server client/server architecture (such as the NT node

10   shown in 202) with transactional client/server architecture (such as the Unisys 2200 enterprise node, as shown at 200).

The environment, as utilized in the present invention, is comprised of several components, including a Monitor 201, a Web Server Extension 237, a plurality of adapters 213, 217, 221, and 207, GGATE 72, a plurality

15   of connectors 229, the WebViewC compiler 290, and a set of libraries 288.

The Monitor 201 functions as the WebTx administrative tool. One function of the Monitor 201 is to direct service requests to the adapters 84, 86, 88, and 207, as appropriate. Within a Unix environment, the monitor module is known as WebMon, while under the Windows NT environment,

20   the WebTx monitor module is known as WtxSvc.

The DCOM server 358 and CORBA server 360 intercept requests intended for WebTx 218, and instead route the requests to the adapters 84, 86, 88, and 207. The DCOM server 358 and CORBA server 360 will also interpret the response from the GGATE 72, and route the reply. The Web

25   Server Extension is connected to the Monitor 201 via interface 203, an HTML requestor component 224 via interface 228, and a Java Applet 226 via interface 234.

The GGATE 72 is a generic gateway which performs tasks which are grouped into conceptual areas. It receives service requests from the adapters and takes whatever action is necessary to fulfill the request. This typically involves transforming a request (such as a URL from a Web

5   Browser or remote procedure calls RPC's from a DCOM client) into a format which is understandable by a Distributed Transaction Processing System such as a Unisys 2200 Enterprise System 200. GGATE 72 also transforms data returned from the Distributed Transaction Processing System 200 into a formatted response which is returned to the requestor.

10   The WebViewC compiler 290 is used in conjunction with specific GGATE requirements, such as ViewGate, TUXGate, and JGate. The WebViewC compiler 290 compiles Open/OLTP view files generated on the OLTP enterprise system to create WebTx view files (.wv) and HTML files (.html). The WebViewC compiler is a free-standing component with no

15   direct communication to any of the other components within the WebTx environment.

Other Components include libraries 288 such as the Software Development Kit (SDK) libraries, which provide framework and functions for building Custom Gateways. The SDK is specifically designed to allow

20   customers to build their own gateways. Another type of library present within the WebTx system are Java Class Libraries, which provide class definitions for building JavaGate compatible applets.

Another tool 290 that may exist as a component is DGateAce. DGateAce is analogous to WebViewC, and is used specifically in conjunction

25   with DGate, as part of the Unisys Pathmate system. DGateAce is further described in a co-pending application entitled, "AN AUTOMATED DEVELOPMENT SYSTEM FOR DEVELOPING APPLICATIONS THAT

16

INTERFACE WITH BOTH DISTRIBUTED COMPONENT OBJECT MODEL (DCOM) AND ENTERPRISE SERVER ENVIRONMENTS".

Unix WebTx uses Inter-Process Communications (IPC) objects such as semaphores, shared memory, message queues and signals, while NT

5 WebTx uses IPC objects such as handles, pipes, mutexes, and events.

FIG. 5 is a functional block diagram of WebTx components utilized within the Microsoft NT environment. This figure shows specific Gateway implementations within the Windows NT node. The SimpleGate Gateway 236 is specifically utilized as a test tool. It merely echoes a request. The

10 TUXGate Gateway 240 provides generalized access to OLTP services through BEA TUXEDO 266. BEA TUXEDO acts as the hub for a distributed enterprise and Internet 3-tier applications. It provides an open environment that supports a wide variety of clients, databases, networks, legacy systems, and communications options. The FileGate Gateway 244 works in

15 conjunction with a specific OLTP service to access textual files on the Unisys 2200 node. ViewGate 248 provides generalized access to OLTP services on the Unisys 2200 note (specifically HTML output). JGate 252 provides generalized Java applet access to OLTP services on the Unisys 2200 node. The DGate Gateway 256 provides generalized DCOM access to

20 OLTP services on the Unisys 2200 node. The MapperGate Gateway 260 provides generalized access to Mapper applications within the Microsoft Windows NT environment. A Custom Gateway, such as shown at 264, provide a way for a customer to build their own Gateway to interface their own applications to an OLTP enterprise application.

25 **FIG. 6** is a diagram showing the relationship of the key software components of WebTx which allow clients to access enterprise applications. The Unisys ClearPath IX Server 310 includes both an OS 2200 environment 312 and a Windows NT environment 314. All ClearPath HMP

IX Servers 310 include On-Line Transaction Processing (OLTP) software that complies with the X/Open model for Distributed Transaction Processing (DTP). This enables client/server access to existing OLTP applications as well as allowing development of new, distributed client/server applications.

5   The X/Open DTP software provided for the OS 2200 environment 312 is TransIT Open/OLTP, available commercially from Unisys Corporation. The TransIT Open/OLTP Transaction Manager 317 is the base product for all Open/OLTP software in the OS 2200 environment 312. It includes: 1) a transaction monitor, which executes and routes transactions, performs

10   load balancing, and recovers transactions after failures and 2) A communications resource manager (CRM), which controls communications among distributed applications.

        The Open/OLTP Transaction Manager 317 includes interfaces to applications and to database systems (resource managers), including the

15   Database Management System (DMS) and the Relational Database Management System (RDMS).

        The OS2200 Environment also includes an Open/OLTP Heritage Application Access component 316. This component 316 allows use of existing OS 2200 OLTP applications, many without modification, as

20   Open/OLTP server programs. This provides an easy way to provide GUI client/server access to existing applications. These Open/OLTP server programs can be Transaction Processing (TIP), High-Volume Transaction Processing (HVTIP) or other online batch programs (as shown at 318). Tools are provided for formatting the data from the existing program into

25   Open/OLTP buffer formats.

        When used with Open/OLTP, the present invention makes it possible to provide access to the following types of OS 2200 applications: 1) Native Open/OLTP applications (local), 2) native Open/OLTP applications that

18

participate in distributed transactions with other platforms running Open/OLTP and BEA TUXEDO software, and 3) Heritage applications that use TIP, HVTIP, and DPS.

Existing transactions can be reused without modification, as long as
5   they meet the following criteria: 1) Open/OLTP services must use the request/response model. Conversational services are not supported, and 2) Open/OLTP services may use X_OCTET buffers.

The key software components that enable clients to access Open/OLTP applications reside in the Windows NT environment 314 of the
10   ClearPath IX server 310.   Connectors 229 consist of middleware which enables applications in the Windows NT environment to execute transactions against OS 2200 applications that use the Open/OLTP transaction manager 317. The GGate runtime software component 322 (GGate.exe) acts as a conduit between the Windows NT DCOM environment
15   314 and the Open/OLTP environment 312. Adapters 324 accept requests from DCOM Clients, repackage the parameters into the format required by the Open/OLTP transaction manager 317, and then forward the parameters over a named pipe to the GGate runtime 322. The DCOM Server 324 could also include a variety of distributed objects. The stub software component
20   326 accepts remote procedure calls from object proxies on client PCs and converts them to interface calls to the DCOM Server application 324.

DCOM Client components 328 reside in a Windows 95 or Windows NT Workstation environment on a personal computer 315. The DCOM Client program 328 provides a graphical user interface (GUI) for submitting
25   transaction requests to Adapters 324 and viewing the data are returned. The Object Proxy software component 330 converts requests from the DCOM client 328 to remote procedure calls (RPC) 334. The RPCs 334 are

subsequently sent across a network 332 to the stub component 326 in the Windows NT environment 314.

FIG. 7 is an illustration of the environment within which the present invention operates for transfers with non-view buffers. Open/ OLTP services 450 are created by a user on an enterprise server 452, such as a Unisys 2200. These services 450 are capable of operating under an OLTP-style transaction processing system. In a preferred embodiment, this OLTP-style system is X/Open compliant. The service 450 is designed to accomplish a specific task, for example, update a user's bank account balance following a debit or credit.

Unlike view based services, services that use pass-through data do not require view files or definitions.

The Distributed Component Object Model (DCOM) is a Microsoft model for distributed object computing. Within the DCOM environment, a remote DCOM Client Application 486 can make a request. The DCOM client 486 can be any type of client, including a Visual Basic client, a C++ client, or a Web Browser with Active Server Pages (ASP). If the request made by the DCOM client 486 is a request for access to a remote process (interprocess request) the request is routed to proxy.dll 484. Proxy.dll 484 is a code segment which receives any client requests targeted for a remote server, and will facilitate the necessary interprocess communications. The proxy.dll 484 understands how to communicate with the Client 486, and also understands how to communicate over an interface 485 which is shared by two or more processes. The proxy.dll 484 "marshals" the request parameters into an independent format so that they may be provided by the client process 486 over the COM-based interface 485, which conforms with the Microsoft DCOM Model. The stub.dll 482, which also understands how to communicate over the common interface 485, "un-marshals" the

parameters into a format that can be understood by the DCOM Server Application.exe 470. Thus, the DCOM environment allows machines with entirely different architectures (PCs, Workstations, etc.) to communicate using a common interface.

5   The specifics of the common interface are described in an Interface Definition Language (IDL) 474. The IDL 474 is operated on by the Microsoft Interface Definition Language (MIDL) compiler 476 to create a set of .c (C) and .h (header) files 478. Then, a second compiler (C++) 480 operates on the .c and .h files to create the stub.dll 482 and the proxy.dll 484. The

10   proxy.dll 484 is linked to the DCOM Client Application.exe 486, and the stub.dll 482 is linked to the DCOM Server Application.exe 470.

Once the DCOM Server 470 un-marshals the parameters, the parameters are packaged into a single buffer and passed to the UDTXNO.dll 468. The UDTXNO.dll 468 and the GGate.exe 466, both of which are the

15   subject of the present invention, are the modules which "DCOM-enable" an OLTP enterprise server 452 such as the Unisys 2200 System.

FIG. 8 shows generalized data flow through the present invention when employing XML message format and view buffers. Client 500 prepares a service request as an XML message. This XML message is sent

20   over the Internet to Adapter 502, which operates as previously described. Xgate prepares the service request buffer from the output of Adapter 502 in accordance with the format described by Input View Definition 510. The service request buffer is provided to connector 506, which transfers it to Server Transaction 508 as described above.

25   The response of Server Transaction 508 to the service request is prepared and transferred to connector 506. It is received as described above. The response arrives as a response buffer which is unpacked by Xgate 504 in accordance with the information from Output View Definition

512. The unpacked response is provided to Adapter 502 wherein it is reformatted as an XML message. The formatted XML response message is transferred to Client 500 via the Internet.

FIG. 9 is a block diagram showing the operation of the components

5　of the present invention within the larger environment without view buffers. In this example, Clients 514, 516, 520, and 522 may each interact with a number of mainframe services via either XML message format (i.e., through Xgate) or a generic gate (through GGate).

Each of the Clients (i.e., 514, 516, 520, and 522) is shown as

10　interacting with a different one of the four Adapters (i.e., 524, 526, 528, and 530) within the example shown. The available adapters shown, without limitation, include: In process DCOM; Out process DCOM; CORBA; and Java RMI. Other formats are possible necessitating other adapters (not shown). As explained above, these adapters provide the formatting

15　necessary to convert the received XML message to a format to interface with either XGate 532 or Ggate 518.

The connectors shown (i.e., HTPic 534, COMAPI 536, and Queue2 538) each provide an interface from Ggate 518 or Xgate 532 to a different legacy mainframe system. As described above, the connectors

20　accommodate the differing formats to these different legacy mainframe systems and enable each to interface with Ggate 518 or XGate 532.

Having thus described the preferred embodiments of the present invention, those of skill in the art will readily appreciate that the teachings found herein may be applied to yet other embodiments within the scope of

25　the claims hereto attached.

WE CLAIM;